Etsi Otto

Android-sovellus

Ammattikorkeakoulututkinnon opinnäytetyö

Riihimäki, tieto- ja viestintätekniikka

2014, 2017

Marko Vehmas

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tieto- ja viestintätekniikka
Riihimäki

Tekijä Marko Vehmas **Vuosi** 2017

Työn nimi Etsi Otto -Android-sovellus

Työn ohjaaja Toni Laitinen

TIIVISTELMÄ

Työn tavoitteena oli luoda Android-kauppapaikalla julkaistavissa oleva sovellus, joka näyttää kartalla kaikki Otto-pankkiautomaatit. Asiakas, Netti-Tv.net-verkkosivuston kehittäjä ja ylläpitäjä, lähestyi minua idean kanssa huomatessaan edellä mainitun tyyppisen sovelluksen puuttumisen kauppapaikalta. Projekti määriteltiin aluksi pääpiirteittäin, ja suunniteltiin ohjelman toimintajärjestys, ja sovittiin asiat, jotka asiakas hoitaa ja toimittaa projektin edetessä.

Projekti onnistui aikataulussa ilman suuria kompromisseja, ja siihen saatiin lisättyä joitakin ominaisuuksia, joita ei alussa suunniteltu. Ohjelmasta tuli hyvä, toiminnallinen paketti, jonka voi häpeilemättä julkaista ja esitellä yleisölle. Android-alustan valinta helpotti ohjelman kehitystä, sillä siinä käytetty Java-ohjelmointikieli oli entuudestaan jokseenkin tuttu, ja Android tukee yhteensopivuutta taaksepäin erittäin pitkälle.

Avainsanat tietotekniikka, Android, sovelluskehitys

Sivut 35 sivua, joista liitteitä 0 sivua

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Bachelor's in Information and Communication Technology
Riihimäki

Author Marko Vehmas **Year** 2017

Subject Etsi Otto -Android application

Supervisor Toni Laitinen

ABSTRACT

The goal of this project was to create a commercial Android application that would list all the ATM-machines in Finland. The commissioner, the developer of Netti-Tv.net, approached me with the idea after noticing an absence of a similar product on the Android applications market place. The project was roughly defined at the beginning, including the app's execution order, and we listed all the issues the commissioner needed to provide as the project progressed.

The project was very successful, being finished on schedule and with some extra features that were not originally planned to be included in the scope of the project. The application ended up being a good, working package, that could be proudly published and presented to the public. The decision to develop the program for the Android operating system helped in that developing for Android is done with the Java programming language which I was familiar with, and the backwards compatibility to older Android versions is impressive.

Keywords information technology, Android, software development

Pages 35 pages including appendices 0 pages

LYHENTEITÄ JA TERMEJÄ

API	Application Programming Interface eli ohjelmointirajapinta, tapa tarjota yhden ohjelman toiminnallisuuksia toiselle. Rajapinnan komennot dokumentoidaan, ja niitä kutsumalla voidaan suorittaa prosesseja rajapinnan toisella puolen, ja vastaanottaa tulokset rajapintaan yhteydessä olevaan prosessiin.
C#	Microsoftin kehittämä oliopohjainen ohjelmointikieli. Pohjautuu Java-ohjelmointikieleen.
CSV	Comma-Separated Values, pilkulla erotetut arvot. Yksinkertainen taulukkotiedostoformaatti, jossa jokainen arvo erotetaan toisistaan pilkuilla, ja pilkutetut arvojoukot sijaitsevat taulukkomaisesti omilla riveillään.
HTTP	Hypertext Transfer Protocol. Protokolla, jota käytetään tiedonsiirtoon internetissä TCP-protokollan päällä. Internetselaimet perustavat toimintansa tähän protokollaan.
Java	Yksi maailman käytetyimmistä oliopohjaisista ohjelmointikielistä. Android pohjautuu Java-kieleen. Kehitetty alun perin verkkoselainympäristöön, mutta on sittemmin levinnyt hyvin erilaisiin käyttöympäristöihin.
JSON	JavaScript Object Notation, erittäin suosittu tiedostoformaatti jossa voidaan tallettaa erilaisia muuttujia JSON-objekteihin.
Olio-ohjelmointi	Suosittu ohjelmoinnin ajattelutapa, jossa erilaiset oliot sisältävät monenlaista tietoa, toimivat yhteistyössä, ja vaikuttavat monipuolisesti ohjelman ajoon. Samasta oliosta on helppo luoda erilaisia versioita eri ominaisuuksin, niitä on mahdollista jatkaa, ja niitä voi olla lukemattomia määriä.
XML	Extensible Markup Language. Merkintäkieli, jolla kuvataan erilaisia rakenteita, kuten tekstitiedoston tai graafisen käyttöliittymän rakennetta. Voidaan merkata esimerkiksi tekstielementti, jolle annetaan erilaisia arvoja, kuten fontti, fontin koko, väri, rivien reunojen tasaus, riviväli.

SISÄLLYS

Ei liitteitä

1 JOHDANTO

Opinnäytetyön aiheena on Android-sovellus, jonka avulla käyttäjä voi nähdä lähellään sijaitsevat pankkiautomaatit dynaamisesti, reaaliaikaisesti, ja mahdollisimman nopeasti. Tarkoituksena on luoda pohja niin, että se olisi helposti muokattavissa asiakkaan halutessa, jolloin ohjelmaa voisi helposti laajentaa tai sitä voisi käyttää pohjana muille, vastaavanlaisille ohjelmille. Suomen pankkiautomaateista (Otto-automaateista) vastaa Automatia Pankkiautomaatit Oy, joka myös tarjoaa kaikkien Suomen pankkiautomaattien sijainnit CSV-muotoisena tiedostona (Automatia, 2017). Tarkoituksena on tätä tiedostoa hyväksikäyttäen piirtää Google Maps -karttaelementille automaatit niin, että käyttäjän sijainti näkyisi myös reaaliajassa, jolloin käyttäjä näkee lähiympäristön automaatit, mutta voi myös halutessaan tutkia minkä tahansa muun alueen pankkiautomaattien sijainteja. Ohjelmasta on siis tarkoitus tehdä käyttäjäystävällinen, käyttäjän tarpeiden mukaan muokattavissa oleva, virheenkestävä, itseään ylläpitävä kokonaisuus.

2 SOVELLUKSEN ALUSTUS JA SELITYS

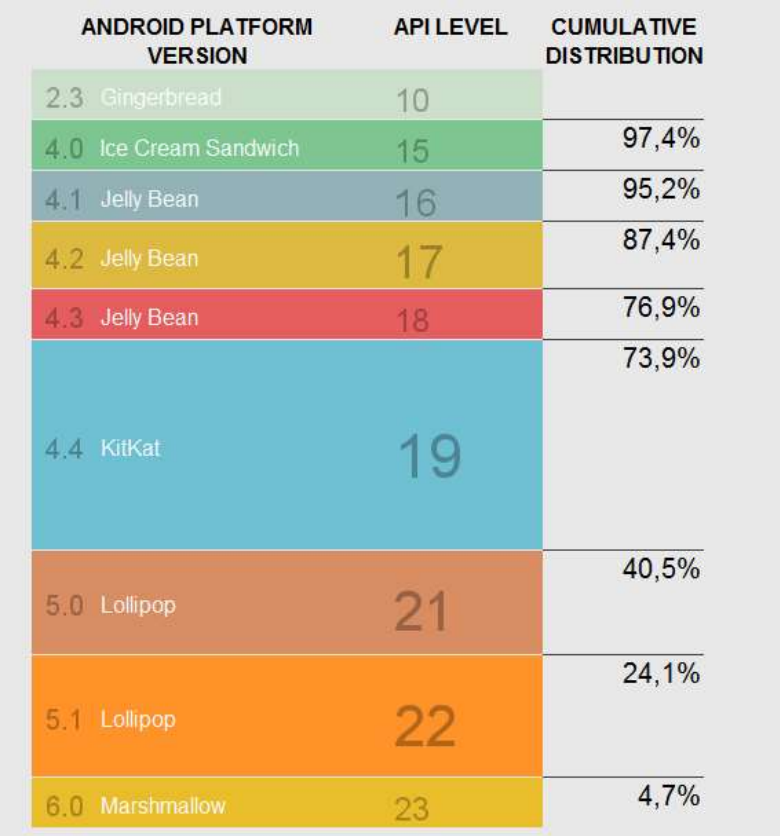
2.1 Työn määrittely

Asiakas tarjoaa käytettäväksi pankkiautomaattilistan JSON-muodossa, joka muunnetaan alkuperäisestä Automatia Pankkiautomaatit Oy:n tarjoamasta CSV-listasta asiakkaan serverillä JSON-muotoon helpomman käsittelyn vuoksi. Nämä pisteet piirrettään Google Maps -kartalle listan sisältämiin koordinaatteihin perustuen. Kartalle piirretään myös käyttäjän sijainti, jonka avulla hän voi nopeasti tarkistaa lähimmät pankkiautomaatit omien tarpeidensa mukaan, ja tarpeen vaatiessa on mahdollista saada reittiohjeet tietylle automaatille nykyisestä sijainnista. Jos asiakas ei anna lupaa käyttää sijaintitietoja, ohjelman tulee silti näyttää automaattien sijainnit.

Ohjelman tulee myös näyttää eri ominaisuuksin varustetut pankkiautomaatit mahdollisimman selkeästi, sekä tarjota käyttäjälle suodattimia joiden avulla käyttäjä voi listata vain tietyntyyppiset automaatit (kuten aukioloajan tai talletusmahdollisuuksien perusteella). Ohjelman tulee kestää myös muutoksia, joita Automatia Pankkiautomaatit Oy tarvittaessa tekee automaattien paikkatietolistaan ilman, että se vaatisi toimia niin käyttäjältä kuin asiakkaaltakaan. Lopullisesta tuotoksesta on tarkoitus saada kaupallinen sovellus, jonka pohjalta asiakas voi jatkaa kehitystyötä halutessaan, eli ohjelmakoodi on kommentoitava ja dokumentoitava asiakkaan toiveiden mukaisesti.

2.2 Android-käyttöjärjestelmä

Android on Googlen kehittämä mobiililaitekäyttöjärjestelmä. Android 1.0 julkaistiin vuonna 2008, ja on sen jälkeen saavuttanut maailmanlaajuisilla markkinoilla valta-aseman. Androidista on kehityksensä aikana julkaistu yli 10 eri versiota, ja Google kerääkin dataa Android-versioiden markkinajakaumasta, ja esittää sen sekä sivuillaan, että osana uuden projektin luontia Android Studiossa. Tämän projektin luontihetkellä Android 4.4 (API 19) ja uudemmat hallitsevat noin 74 % maailmanlaajuisista markkinoista (Kuva 1). Koska ohjelma tulee olemaan käytettävissä vain Suomen alueella, eikä ole mahdollista nähdä käyttöjärjestelmäversioiden jakaumaa alueittain, syntyy pieni ongelma. Voidaan kuitenkin olettaa, että Suomessa Android 4.4 vanhempia versioita käyttäviä laitteita löytyy prosentuaalisesti huomattavasti vähemmän kuin maailmalla, joten yli 75 % suomalaisista Android-laitteiden käyttäjistä on sekin suuri potentiaalinen kohdeyleisö. Tämän vanhempien Android-versioiden tukeminen mahdollisesti rajoittaisi ohjelman toimintoja, sekä toisi erittäin pienen lisän käyttäjäkuntaan, joten tyydytään API:in 19 ja uudempiin versioihin.



Kuva 1. Android-versioiden markkinajakauma alkuvuodesta 2017 (Google, 2017-a).

2.3 Google Play

Google Maps ja Location ovat osa Google Play -palvelupakettia (Google Play services package). Google aloitti tämän käytön vuonna 2012, jolloin

eri Play-palveluiden käyttö kolmannen osapuolen sovelluksissa haluttiin tehdä helpoksi, sekä mahdollistaa kaikkien Android-laitteiden päivittämistä ilman ongelmia eri laitevalmistajien kanssa (Wikipedia, 2017-a). Paketti on kuitenkin suuri, ja harvoissa sovelluksissa tarvitaan kaikkia tarjottavia lisäominaisuuksia. Siksipä marraskuussa 2014 Google muutti paketin käyttöä versiosta 6.5 eteenpäin niin, että siitä voidaan lisätä sovellukseen vain haluttuja ominaisuuksia (Google Developers, 2017-a).

Palvelupakettia päivitetään pitkälle taaksepäin, jotta hyvinkin vanhat Android-versiot saavat uusimmat palvelupaketin ominaisuudet, jolloin paketin käyttö ei rajoita ohjelman sopivuutta vanhempien käyttöjärjestelmien kanssa. Ohjelman voi lisäksi määrätä käyttämään aina uusinta palvelupaketin versiota, mutta se tarkoittaisi ohjelman mahdollista rikkoutumista aina, kun palvelupaketti päivitetään. Siksi on suositeltavaa määrätä ohjelma käyttämään tiettyä paketin versiota. Tämän kirjoitushetkellä uusin palvelupaketin versio on helmikuussa 2017 julkaistu versio 10.2. Tämä on ensimmäinen versio jossa Android 2.3-version tuki päättyi (Google Developers, 2017-b). Android 2.3 -version markkinaosuus on alle 2 %, joten tämän version käyttö tarkoittaisi marginaalisen pienen, Suomessa mahdollisesti olemattoman asiakaskunnan menetystä. Tämän vuoksi on mahdollista käyttää tällä hetkellä uusinta palvelupaketin versiota, ja näin valita tietyt ohjelman käyttämät paketin ominaisuudet, kuitenkaan käyttäjiä menettämättä.

2.4 **JavaScript Object Notation (JSON)**

JSON on avoimen standardin tiedostoformaatti, jossa selkolukuisesta tekstistä muodostetaan dataobjekteja käyttäen objekteja (object) jotka muodostuvat avain-arvo-pareista, sekä listoja (array). JSON-tiedostojen keveys, helppolukuisuus, kieliriippumattomuus ja helppo syntaksi ovat auttaneet tekemään siitä yhden suosituimmista dataformaateista, ja sen suosion vuoksi monet ohjelmointikielet tukevatkin JSON-formaatin luontia ja lukemista. Kuvissa 2 ja 3 näytetään ja vertaillaan CSV-muotoisen ja JSON-muotoisen tiedon tallettamista. CSV-lista muistuttaa rakenteeltaan vahvasti taulukkoa. JSON-objektit taas ovat olioiden kaltaisia, ja niiden muokkaaminen, linkittäminen, ja niihin viittaaminen on helppoa ja selkeää. Tässä työssä käytetään JSON-objekteja pankkiautomaattien tietojen varastoimiseen. Jokainen automaatti on yksi objekti, ja ne sisältävät samat tiedot, eli avaimet. Näitä tietoja hyödyntämällä projektissa luodaan kartalle saman näköisiä merkkejä eri tiedoin. (Wikipedia, 2017-b.)

```
1  First name,Surname,Age,Sex,Married,Children
2  John,Doe,35,Male,No,1
3  Jane,Doe,29,Female,Yes,0
4
5  {
6  "First name":"John",
7  "Surname":   "Doe",
8  "Age":       35,
9  "Sex":       "Male",
10 "Married":   false,
11 "Children":  0
12 }
13
14 {
15 "First name":"Jane",
16 "Surname":   "Doe",
17 "Age":       29,
18 "Sex":       "Female",
19 "Married":   true,
20 "Children":  0
21 }
22
```

Kuva 2. Riveillä 1-3 tiedot talletettuna CSV-muodossa. Riveillä 5-21 samat tiedot talletettuna JSON-muodossa.

```
1  First name,Surname,Age,Sex,Married,Children
2  John,Doe,35,Male,No,1
3  Jane,Doe,29,Female,Yes,0
4
5  var person1 = {
6      "First name":"John",
7      "Surname":   "Doe",
8      "Age":       45,
9      "Sex":       "Male",
10     "Married":   false,
11     "Children":  [person2, person3]
12 }
13
14 var person2 = {
15     "First name":"Jane",
16     "Surname":   "Doe",
17     "Age":       14,
18     "Sex":       "Female",
19     "Married":   false,
20     "Children":  0
21 }
22
23 var person3 = {
24     "First name":"Jack",
25     "Surname":   "Doe",
26     "Age":       14,
27     "Sex":       "Male",
28     "Married":   false,
29     "Children":  0
30 }
31
```

Kuva 3. Huomaa henkilön lapsien talletus rivillä 11. JSON-objektiin on talletettu luettelo, joka koostuu kahdesta muusta JSON-objektista.

3 SOVELLUKSEN KEHITYS

3.1 Kehitysympäristön valmistelu

Ohjelma kehitetään mobiililaitteille, joten vaihtoehtoja kehitysympäristöissä on useita. Tarkoitus on kuitenkin tehdä ohjelmasta vain Android-versio, joten paras tuki kehitykseen löytyy varmasti Androidia kehittävän Googlen omasta Android Studio -kehitysympäristöstä. Jos ohjelmaa kehitettäisiin muille käyttöjärjestelmille sopivaksi, hyvä vaihtoehto kehitykseen olisi esimerkiksi Xamarin, jossa C#-kielellä tehty ohjelma voidaan automaattisesti kääntää lähes kokonaan sekä Androidille, iOS:lle että Windowsille (Xamarin, 2017). Tämän työn asettamien vaatimusten vuoksi työssä käytetään kuitenkin Android Studiota, ja työ tehdään Androidille natiivilla Java-ohjelmointikielellä.

Jokainen ohjelma (app) Android Studiossa on oma projektinsa, ja ohjelmaan saadaan oikeanlainen pohja luomalla uusi projekti ja määrittelemällä se oikein. Projektin luonti alkaa sen nimellä, tässä tapauksessa jo ennalta sovittu "Etsi Otto". Ohjelman domain tulee myös asiakkaalta. Siihen voidaan lisätä C++-tuki haluttaessa, mutta tässä tapauksessa ei ole ainakaan ennalta tiedossa mitään C++-kirjastoja, joita ohjelman kehitys vaatisi, joten C++-tuki voidaan jättää lisäämättä (Android Developers, n.d.-a).

Ennen eteenpäin siirtymistä voidaan valita vielä hakemisto, johon projekti luodaan. Seuraavaksi on määriteltävä kuinka laajan alustatuen ohjelman pitäisi saada, ja tässä on huomioitava muutama asia. Mitä vanhempia käyttöjärjestelmiä ohjelma tukee, sitä laajemman yleisön se saavuttaa. Toisaalta, vanhemmat käyttöjärjestelmät rajoittavat ohjelman käytössä olevien ominaisuuksien määrää, jolloin liian vanhat käyttöjärjestelmät voi vaikeuttaa sen kehitystä, tai tehdä siitä jopa mahdottoman. Teoriaosuudessa määritettiin jo kohdeversioksi sopivan versio 4.4, eli API 19.

Samalla määritellään projektin kohdelaitteet, jotka tässä tapauksessa ovat sekä älypuhelimet että tabletit. Seuraavaksi valitaan, minkälainen aktiviteetti ohjelmasta halutaan luoda. Kehityksen vapauden takia helpoin lähtökohta on tyhjä aktiviteetti (empty activity), jolloin lähdekoodiin tulee mahdollisimman vähän ylimääräistä. Annetaan aktiviteetille ja sen muotoilutiedostolle nimet, ja annetaan Android Studion luoda projekti aikaisemmin määriteltyyn kohdehakemistoon. Kun projektin luonti on valmis, löytyy projektin hakemistoista muutama tiedosto, joista tärkeimpiä ovat ohjelman pääaktiviteetin lähdekoodin sisältävä java-tiedosto, pääaktiviteetin

muotoilutiedot (layout) sisältävä xml-tiedosto sekä ohjelman manifestitiedosto, joka sisältää perustietoja ohjelmasta, kuten sen käynnistyskuvakkeen polun, ja ohjelman nimen. Projektin pohja on nyt luotu valmiiksi, ja voidaan aloittaa ohjelman luonti.

3.2 **Pankkiautomaattien tietotiedoston nouto**

Aluksi pitää noutaa konvertoitu tietotiedosto asiakkaan palvelimelta internetin yli. Tätä varten ensimmäinen askel onkin internetyhteyden tarkistaminen. Android-ohjelma ei saa lupia käyttää kaikkia puhelimen ominaisuuksia oletuksena, vaan luvat eri ominaisuuksien hyödyntämiseen pitää pyytää käyttäjältä erikseen. Tähän on kaksi eri tapaa: API 22 tai vanhemmat (Android 5.1 ja aikaisemmat) versiot pyytävät luvat eri ominaisuuksien käyttöön ohjelmaa asennettaessa, ja API 23 ja uudemmat (Android 6.0 ja myöhemmät) versiot tukevat lupien pyytämistä kesken ohjelman suorituksen, jolloin käyttäjän on helpompi ymmärtää ominaisuus jolle hän on antamassa lupaa, ja mihin tarkoitukseen kyseisiä ominaisuuksia ohjelmassa tarvitaan.

On myös mahdollista näyttää käyttäjälle selitys ominaisuuden käyttötarkoituksesta kesken ohjelman käytön, sen sijaan että kaikkien ominaisuuksien käyttötarkoitukset selitettäisiin ohjelmaa ladattaessa yhdessä, pitkässä tekstidokumentissa. Lupien erillinen kysyntä mahdollistaa myös yksittäisten lupien myöhemmän sallimisen ja kieltämisen ohjelman asetuksista käyttöjärjestelmän valikossa, toisin kuin ennen API 23:a, jolloin eri lupien kieltäminen asennuksen jälkeen onnistui vain poistamalla ohjelma. (Android Developers, n.d.-b.)

Eri lupien pyytämiseen tarvitaan vähintään kaksi lisäystä ohjelman tietoihin: ohjelman manifestitiedostoon on lisättävä halutun ominaisuuden nimi, ja ohjelmakoodiin on lisättävä pyyntökomento, jota kutsutaan halutulla hetkellä. Aloitetaan lisäämällä manifestitiedostoon internetyhteyden käyttölupa, ja myöhemmin lisätään ohjelmakoodiin ajon aikainen käyttölupapyyntö. Androidilla internetyhteyteen liittyvät kysymykset ja testit suorittaa ConnectivityManager-luokka, joka palauttaa vastauksen NetworkInfo-muodossa. NetworkInfo-muotoisesta palautteesta voidaan testata, onko internetyhteys toiminnassa (Android Developers, n.d.-c). Internetyhteyden käyttölupa kuuluu Androidilla niin sanottujen "normaalien" lupien piiriin, eli sen käyttöön ei tarvitse pyytää lupaa erikseen (Android Developers, n.d.-b). Tarkistetaan vain aluksi, että laite on yhteydessä verkkoon, ja jos on, noudetaan tietotiedosto asiakkaan serveriltä paikallista käsittelyä varten.

Hyvin usein, kun jokin ohjelma suorittaa prosessia (kuten laskentaa tai tietojen noutoa), on tärkeää miettiä miten pitkään prosessi kestää, ja kannattaisiko se ajaa tausta-ajona. Jos ohjelma tekee jotain pitkään, sitä ei ole suositeltavaa tehdä samassa säikeessä (thread) kuin käyttöliittymää, sillä kyseinen iso prosessi voi varata koko säikeen, jolloin käyttöliittymälle ei jää

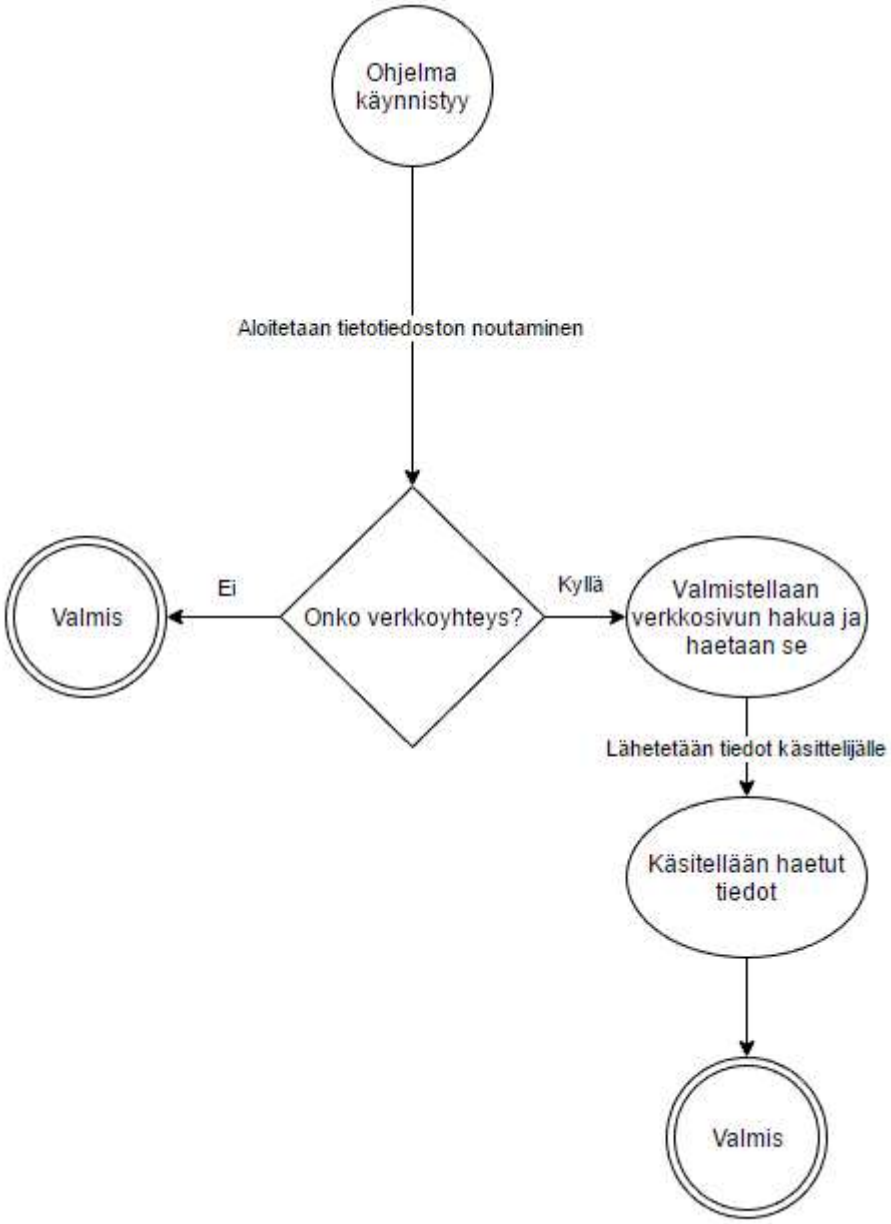
prosessoriaikaa lainkaan, ja käyttöliittymä jäätyy. Tästä johtuen käyttäjä saattaa luulla ohjelman kaatuneen ja sulkee sen. Siksi tämäkin tietojen haku ja prosessointi, jonka tiedetään kestävän vähintään muutamia sekunteja, halutaan ajaa taustalla. Tällöin käyttäjä voi jatkaa ohjelman käyttöä ja ohjelmassa voidaan informoida käyttäjälle, että tämä joutuu odottamaan hetken, kun tiedostoa valmistellaan. Androidilla tausta-ajoa varten on suunniteltu AsyncTask-luokka, jota jatkamalla voidaan tehdä omia luokkia ja toimintoja, jotka tapahtuvat käyttöliittymästä erillisessä säikeessä. AsyncTaskissa on huomioitava, että se on suunniteltu käytettäväksi vain muutaman sekunnin mittaisten taustaprosessien ajamiseksi, ja pidempikestoisten prosessien ajamiseksi on suositeltavaa käyttää Executors-luokkaa. (Android Developers, n.d.-d). Tietotiedoston haku ei kuitenkaan kestä huomattavia aikoja sen suhteellisen pienen koon vuoksi, joten AsyncTask sopii tehtävään ainakin toistaiseksi. AsyncTaskista käytän kahta metodia: ensimmäinen hakee taustalla halutun tiedon, ja toista kutsutaan, kun ensimmäinen metodi on suorittanut työnsä loppuun.

AsyncTaskin ensimmäistä metodia, doInBackground(), kutsutaan ensin, ja se määritetään hakemaan haluttu tieto (Android Developers, n.d.-d). Tieto esitetään verkkosivulla, joten sen voi noutaa HTTP-yhteydellä lukemalla asiakkaan verkkosivun koko lähdekoodin, joka koostuu vain yhdestä JSON-tiedostosta, ja käsitellä sitä ohjelmassa tekstimuodossa (string). Luodaan siis HttpURLConnection-tyypin muuttujan, jolle määritetään, kuinka kauan yhteys voidaan pitää päällä ja mitä HTTP-metodia käytetään. On määriteltävä myös, halutaanko käyttää yhteyttä tietojen hakemiseen vai syöttämiseen. (Android Developers, n.d.-e). Kun yhteys on valmisteltu, tallennetaan sen hakema tieto InputStream-muotoiseen muuttujaan. Tämä muuttuja käsitellään BufferedReader- ja StringBuilder-toiminnoilla, jotka yhdessä lukevat saadun tiedon, ja tallettavat sen tekstimuodossa. Lopuksi HTTP-yhteys suljetaan, ja siirrytään AsyncTaskissa eteenpäin. (Kuva 4.)

```
private String ConnectToWebPage (String myurl) throws IOException {
    InputStream websiteInputStream = null;
    try {
        URL targetUrl = new URL(myurl);
        //Luodaan yhteysobjekti haluttuun URL-osoitteeseen
        HttpURLConnection httpUrlConnection = (HttpURLConnection) targetUrl.openConnection();
        //Kuinka kauan sivun lukemista suoritetaan max (ms)
        httpUrlConnection.setReadTimeout(15000);
        //Kuinka pitkään yhteyttä pidetään yllä max (ms)
        httpUrlConnection.setConnectTimeout(20000);
        //Käytetään HTTP GET-metodia
        httpUrlConnection.setRequestMethod("GET");
        //Noudetaan verkkosivun sisältö (getInputStream() luo yhteyden itsestään)
        websiteInputStream = httpUrlConnection.getInputStream();
        //Lopuksi palautetaan saatu merkkijono InputStream-muodossa
        return WebpageSourceToString(websiteInputStream);
    } finally {
        if (websiteInputStream != null) {
            //Kun kaikki teksti on haettu, sammutetaan InputStream
            websiteInputStream.close();
        }
    }
}
```

Kuva 4. HttpUrlConnection-yhteysmuuttujan määrittäminen ja käyttö.

AsyncTaskin toista metodia, onPostExecute(), kutsutaan, kun AsyncTaskin muut prosessit ovat valmiita. Tämä metodi suoritetaan käyttöliittymälle varatulla säikeellä, jota tarvitaan, jotta voidaan lisätä käyttöliittymään myöhemmin lisättävään Maps-elementtiin merkkejä, jotka vastaavat pankkiautomaatteja. (Android Developers, n.d.-d.) Siispä onPostExecute()-metodia kutsutaan vasta, kun koko tietotiedosto on noudettu ja muutettu tekstimuotoon, jolloin päästään jäsentelemään haettu tieto. Tieto saadaan JSON-muodossa: koko tiedosto on yksi lista (array), jossa jokainen automaatti on oma objektinsa. Jokainen objekti sisältää samoin nimettyjä tietoja, kuten sijainnin, aukioloajat, ja automaatin mallin. Näitä tietoja tallennetaan eri muuttujiin, joiden avulla rakennetaan myöhemmin kartalle merkkejä, joihin sijoitetaan aina tietyn automaatin tiedot. Tässä vaiheessa tärkeintä on saada tiedot näkyviin, ja testattuani ohjelma, sain ennalta sattumanvaraisesti valitsemieni automaattien tiedot näkyviin konsoliin, eli ohjelma hakee tietotiedoston ja tulkitsee sen onnistuneesti (Kuva 5). Seuraavaksi tavoitteena on lisätä ohjelman toinen keskeinen elementti, eli kartta



Kuva 5. Ohjelman toimintajärjestys tässä vaiheessa.

3.3 Google Maps -karttaelementti

Aluksi ohjelmaan lisätään käyttöön Google Maps -karttaominaisuus. Maps APIa käyttääkseen ohjelma tarvitsee ohjelmakohtaisen Maps API-avaimen, jonka saa generoitua ilmaiseksi, ja jonka avulla voidaan seurata ohjelman käyttömäärää. API-avain onkin ei-julkinen merkkisarja, jolla Google tunnistaa juuri tietyn ohjelman. Avaimen luontiin Google tarjoaa pikaisen prosessin. Kaikkia Google API:a käyttäviä projekteja voidaan hallinnoida yhdeltä sivulta osoitteessa console.developers.google.com, joten sinne luodaan uusi projekti. Projektille annetaan nimi, ja se lisätään järjestelmään. Sitten siihen lisätään Maps-API, ja generoidaan sille avain. Hallinnointisivulta voidaan seurata esimerkiksi avaimen tekemiä pyyntöjä API:lle tai virheiden määrää, eli voidaan seurata käyttäjien aktiivisuutta ja ohjelman toimivuutta.

GMAPI-avain lisätään projektin manifestitiedostoon, ja päivitetään samalla manifestin lupalistaa, eli lisätään ohjelman tietoihin, että se käyttää Maps-karttoja. Käyttäjän sijaintitietojen lupatiedot lisätään myöhemmin. Google Play -palvelupaketti sisältää internetyhteyden lupatiedot, eli sitä ei tarvitsisi lisätä lupalistaan, mutta se lisätään selvyyden vuoksi. Nyt karttaominaisuudet on lisätty ohjelman tietoihin, joten voidaan lisätä karttaelementti myös käyttöliittymään ja ohjelmakoodiin.

Käyttöliittymän muokkaukseen Android Studio tarjoaa kaksi rinnakkain käytettävää keinoa; XML-kielellä rakennetun layout-tiedoston manuaalinen muokkaus, tai käyttöliittymän rakentaminen graafisen kehitysympäristön kautta. Muutokset yhdessä peilautuvat myös toiseen, jolloin käyttöliittymään tehdyt muutokset XML-tiedostossa näkyvät heti graafisessa kehitysympäristössä, eikä ohjelmaa tarvitse ajaa testiympäristössä käyttöliittymämuutosten tarkastelemiseksi.

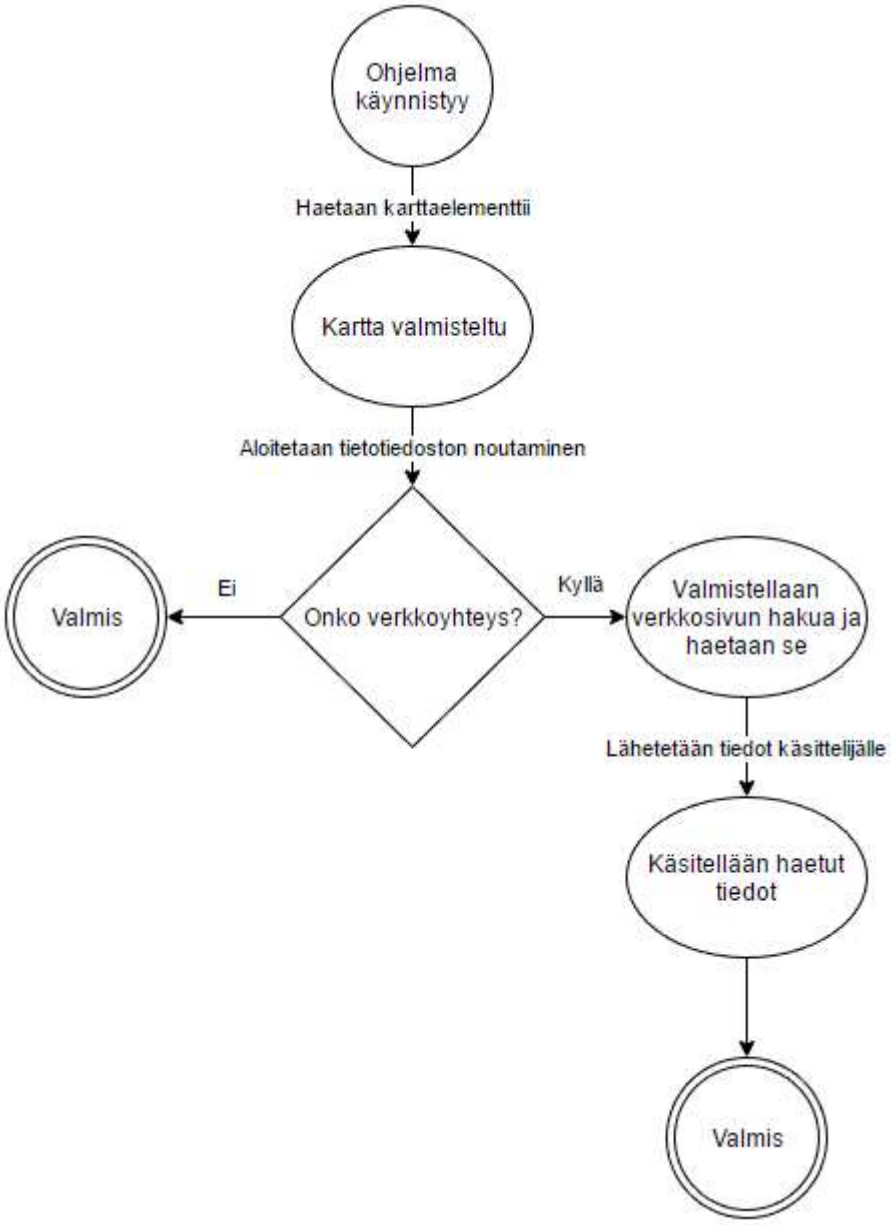
Tällä hetkellä käyttöliittymä on tyhjä, joten siihen lisätään Fragment-objekti, joka tarkennetaan MapFragment-elementiksi. Jotta ohjelmaan ei tarvitse luoda karttaluokkaa erillisenä tiedostona, määritetään MainActivity-luokka implementoimaan OnMapReadyCallback-toiminnallisuus, jolloin MainActivity-luokkaan saadaan karttatuki (Android Developers, n.d.-f). Lisätään ohjelmakoodiin myös muuttuja karttaelementille, ja ohjelmaa käynnistettäessä kutsuttavassa OnCreate-funktiossa yhdistetään käyttöliittymän karttaelementti ohjelmakoodin karttaelementtimuuttujaan. Tämä linkki luomalla mahdollistetaan karttaelementin käsitteleminen ohjelmakoodissa. Lisätään MainActivity-luokkaan nyt OnMapReady()-funktio, johon voidaan sijoittaa koodi, jonka halutaan ajettavan, kun karttaelementti on valmis käytettäväksi. (Kuva 5.)

Karttaelementin testaamiseksi määritellään kartan kameran uudet asetukset niin, että kartta siirtyy Suomen kohdalle, kun kartta on muuten valmis-

teltu. OnMapReady-funktion kutsumiseksi pitää lopuksi määrätä kartta-muuttuja kutsumaan getMapAsync()-funktiota. Testauksessa kartta siirtyi Suomen yläpuolelle, joten kartta on nyt valmisteltu käytettäväksi (Kuva 6).

Kuva 6. Kartan valmistelun yhteydessä kamera siirtyy ennalta määrättyihin koordinaatteihin.

Seuraavaksi yhdistetään jo tehdyt ominaisuudet, eli määritellään pankkiautomaatit näkymään yksittäisinä pisteinä kartalla.

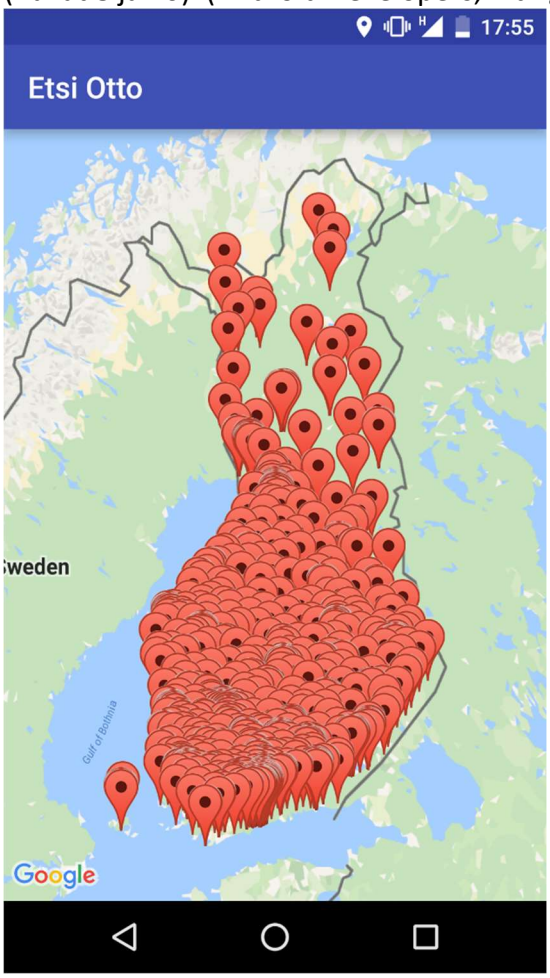


Kuva 7. Ohjelman toimintajärjestys tässä vaiheessa.

3.4 **Automaattien näyttäminen kartalla**

Nyt informaatio saadaan haettua ja jäsenneltyä oikein, ja kartta on valmiina käytettäväksi, joten niitä voidaan lähteä yhdistämään, eli piirretään kartalle automaattien sijainnit. Automaatit saadaan kartalle helpoiten käyttämällä karttakomponenttiin rakennettuja marker-merkkejä. Tuodaan ensin tuki merkeille projektiin, ja lisätään merkkejä kutsumalla addMarker()-funktiota karttamuuttujalla. Funktio ottaa sisään yhden argumentin, joka kertoo merkin ominaisuudet, ja se voidaan rakentaa MarkerOptions()-funktiolla. Merkeille voidaan määritellä muun muassa sijainti, otsikko, lisätekstiä, väri, kuvake, kulma, ja näkyvyys. Testimerkiksi kuitenkin riittää yksi merkki, jolla on sijainti ja otsikko. Kun merkin toimivuus on varmistettu,

luodaan funktio, joka tekee saman käyttäen tietotiedostosta saatavia automaatin koordinaatteja ja muita tietoja, ja populoidaan kartta merkeillä (Kuvat 8 ja 10). (Android Developers, n.d.-g.)



Kuva 8. Kaikki automaatit kartalle piirrettyinä pisteinä.

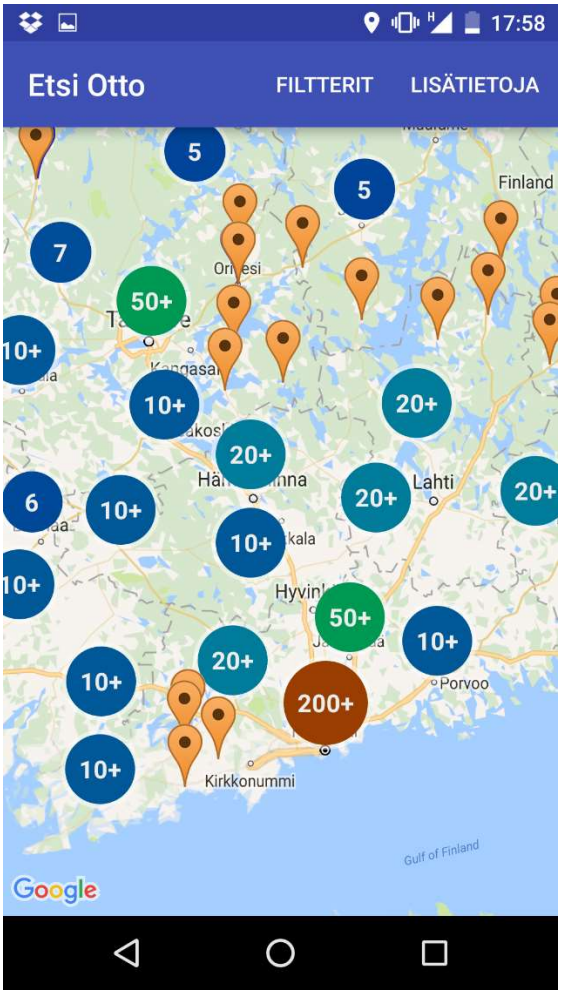
Automaatteja on Suomessa kuitenkin yli 1390 automaattipistettä, ja niiden kaikkien näyttäminen samaan aikaan vaatii paljon enemmän laskentatehoa kuin mobiililaitteissa on tarjolla, ja vaikka se olisikin saatavilla, kaiken sen laskentatehon käyttäminen olisi tarpeetonta. Googlen tarjoaakin vaihtoehdoksi usean merkin samanaikaiselle näyttämiselle ryhmittyvät merkit (clusterable marker), jotka yhdistyvät toisiinsa dynaamisesti riippuen siitä, kuinka monta merkkiä niiden läheisyydessä on samanaikaisesti. Tätä hyödyntämällä onnistutaan vähentämään samanaikaisesti piirrettävien kohteiden määrää, ja näin nopeutettua ohjelman toimintaa.

Lisätään projektiin tuki ryhmittyville merkeille gradle.build-tiedostoon sekä ohjelmakoodiin. Projektiin pitää luoda merkeille oma luokka, joka tukee merkkien ryhmittelyä ja jotka liitetään ryhmittelyjohtajaan (cluster manager), joka hoitaa yhdistelyn valmiiden algoritmien perusteella. Merkkien ryhmittely riippuu niiden läheisyydestä toisiinsa, joka taas riippuu kartan kameran korkeudesta; mitä kauempana maanpinnasta kamera on, sitä

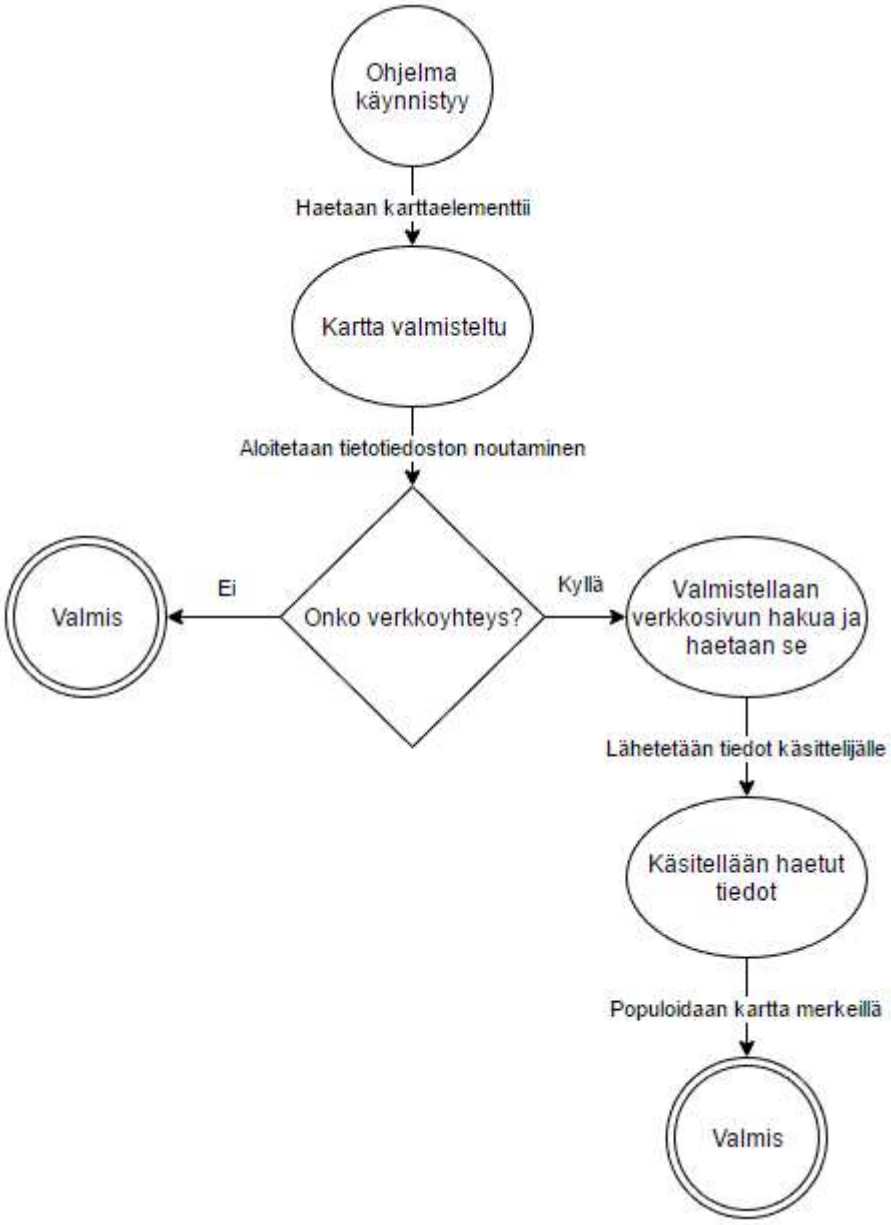
useampi merkki ruudulla näkyy ja sitä enemmän niitä pitää yhdistellä. (Google Developers, 2017-c.)

Koska yhdistely riippuu kameran korkeudesta ja sijainnista, laskenta suoritetaan silloin kun käyttäjä ei liikuta kameraa. Jos laskenta suoritettaisiin kameraa liikutettaessa, se hidastaisi kameran liikkumista ja kuluttaisi resursseja turhaan laskiessaan merkkejä uudestaan ja uudestaan kameran sijainnin muuttuessa. Siksi laskenta suoritetaankin vain kameran ollessa paikallaan. Tätä varten kartassa on CameraIdleListener-kuuntelijakomponentti, joka kuuntelee kartan kameran siirtoja. Kun käyttäjä ei siirrä kameraa tarpeeksi pitkään aikaan, kuuntelija antaa luvan hoitaa merkkien yhdistelylaskennan. Jos käyttäjä aloittaa kameran liikuttamisen kesken laskennan, laskenta pysäytetään ja jäädään odottamaan uutta lupaa aloittaa yhdistelylaskenta.

Lisätään ryhmittelyjohtaja CameraIdleListener-komponenttiin, ja pyydetään ryhmittelyä, kun kaikki merkit ovat valmiita, jolla varmistetaan, että merkit tulevat näkyviin, kun tietotiedosto on jäsennelty ja merkit luotu. Nyt merkit näkyvät kartalla käyttäjälle samoin, kuin jos ne olisi tehty normaaleina marker-merkkeinä, mutta ryhmittymisestä johtuen merkit eivät hidasta ohjelmaa käyttökelvottomaksi aina kameran ollessa kaukana maan pinnasta (Kuva 9). Uusia merkkejä luodessa tulkitaan tietotiedostosta automaattien tyyppejä, ja jaetaan merkit kahteen ryhmään: oranssit merkit ovat automaatteja, joilta voi nostaa rahaa, ja siniset merkit ovat automaatteja, joilla voi nostamisen lisäksi myös tallettaa rahaa.



Kuva 9. Erikokoisia merkkiryhmiä, sekä yksittäisiä merkkejä.



Kuva 10. Ohjelman toimintajärjestys tässä vaiheessa.

3.5 **Käyttäjän näyttäminen kartalla**

Ohjelman käynnistyessä eri komponenttien toimintajärjestys olisi lopussa tarkoitus olla lyhyesti seuraava: tarkistetaan käyttäjän sijainnin saatavuus, haetaan tietolista, piirretään automaatit kartalle, piirretään käyttäjä kartalle, ja zoomataan käyttäjän sijaintiin, jos se nähdään. Käyttäjän sijaintitietoja ei ole vielä ohjelmassa, joten lisätään se seuraavaksi. Käyttäjän sijainnin näkemiseksi käyttäjältä pitää saada lupa sijainnin lukemiseen. Sijainti saadaan toki puhelimesta, mutta tiedot siitä menevät Googlelle, josta sitä voidaan kysellä GoogleApiClient-komponentilla. GoogleApiClient on kokoelma ohjelman tarvitsemia palveluita, jotka yhdessä kysyvät tietoja

Googlen API:lta (Google Developers, 2017-d). Tässä ohjelmassa ainoa tarvittava palvelu on LocationServices, eli paikkatietopalvelu.

Jälleen ensimmäiseksi lisätään projektin manifestiin tieto siitä, että projekti hyödyntää sijaintitietoja, eli lisätään projektin koontitietoihin play-services-location. Tuodaan projektiin GoogleApiClient, ja käytetään siinä olevaa rakentajaa luomaan ohjelmalle oma GoogleApiClient. GoogleApiClient-rakentaja tarvitsee muutaman parametrin: addConnectionCallbacks, joka kertoo koska yhteys API:in syntyy tai pysähtyy, addOnConnectionFailedListener, joka kertoo kun yhteysyritys epäonnistuu, sekä tarvittavat API:t, joita on tässä tapauksessa vain yksi: LocationServices. Tämän jälkeen GoogleApiClient rakennetaan, ja säilötään muuttujaan. Aina ohjelman aktivoituessa tarkistetaan, onko muuttuja tyhjä, ja jos on, luodaan GoogleApiClient. Jos muuttujassa on säilöttynä jo GoogleApiClient, ei sitä tarvitse luoda uudelleen (kuten esimerkiksi silloin, kun näyttö sammutetaan, ja laitetaan takaisin päälle). (Android Developers, n.d.-h.) (Kuva 11.)

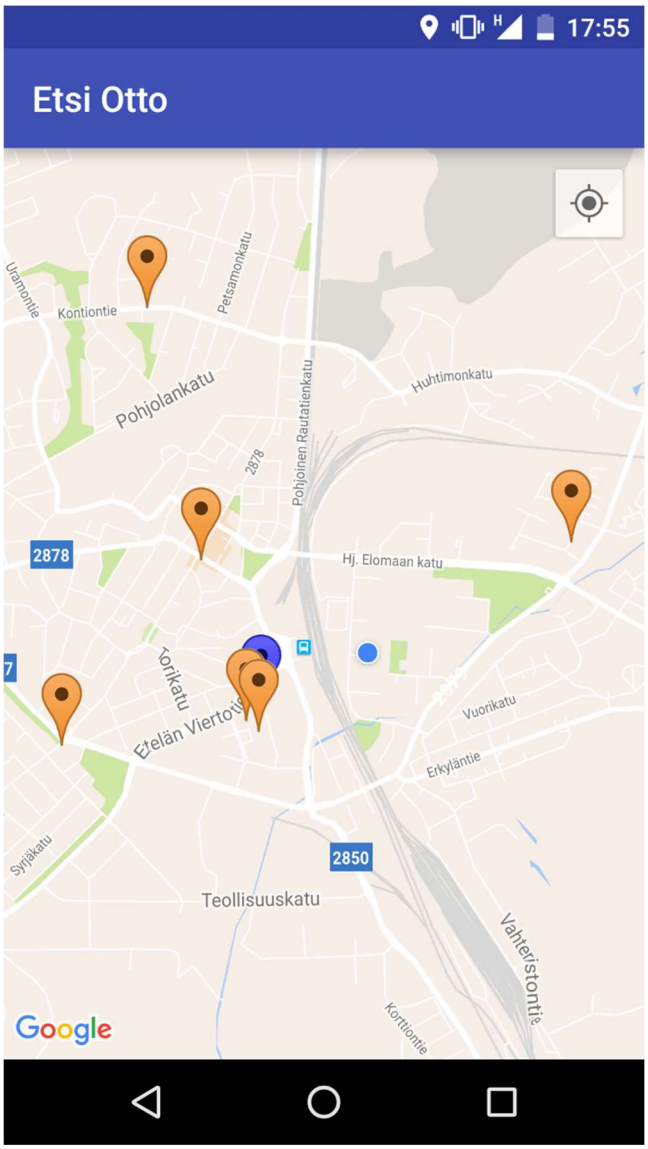
```
    if (googleApiClient == null) {
        googleApiClient = new GoogleApiClient.Builder(this)
                .addConnectionCallbacks(this)
                .addOnConnectionFailedListener(this)
                .addApi(LocationServices.API)
                .build();
    }
}

protected void onStart () {
    googleApiClient.connect();
    super.onStart();
}
```

Kuva 11. Google API Clientin valmistelu, ja siihen yhdistäminen ohjelman palatessa aktiiviseksi onStart-funktiossa.

Kun yhteys Googlen rajapintaan saadaan, voidaan aloittaa käyttäjän sijainnin näyttäminen kartalla. Kun pyyntö sijainnin näyttämiseksi tulee, tarkistetaan ensin, onko sijaintia lupa käyttää. Jos käyttäjä ei halua jakaa sijaintiaan, ei sitä voida myöskään näyttää. Jos käyttäjä taas haluaa nähdä sijaintinsa kartalla, saadaan sijaintitiedot LocationServices.FusedLocationApi.requestLocationUpdates-komennolla. Komentoa kutsuttaessa sille pitää antaa ohjelman GoogleApiClient, locationRequest-asetuskokoelma sekä locationListener-kuuntelija. LocationRequest sisältää tietoja, kuten päivitysten aikavälin ja sijainnin tarkkuuden. LocationListener on kuuntelija, jossa olevaa onLocationChanged-komentoa kutsutaan aina käyttäjän sijainnin päivittyessä.

Sijainnin tarkkuutta voidaan mitata kahdella tasolla: mahdollisimman tarkka, resursseja enemmän kuluttava FINE_LOCATION, sekä epätarkempi mutta kevyempi COARSE_LOCATION. Tarkka sijainti näyttää käyttäjän sijainnin muutaman metrin tarkkuudella, kun taas epätarkempi näyttää sen

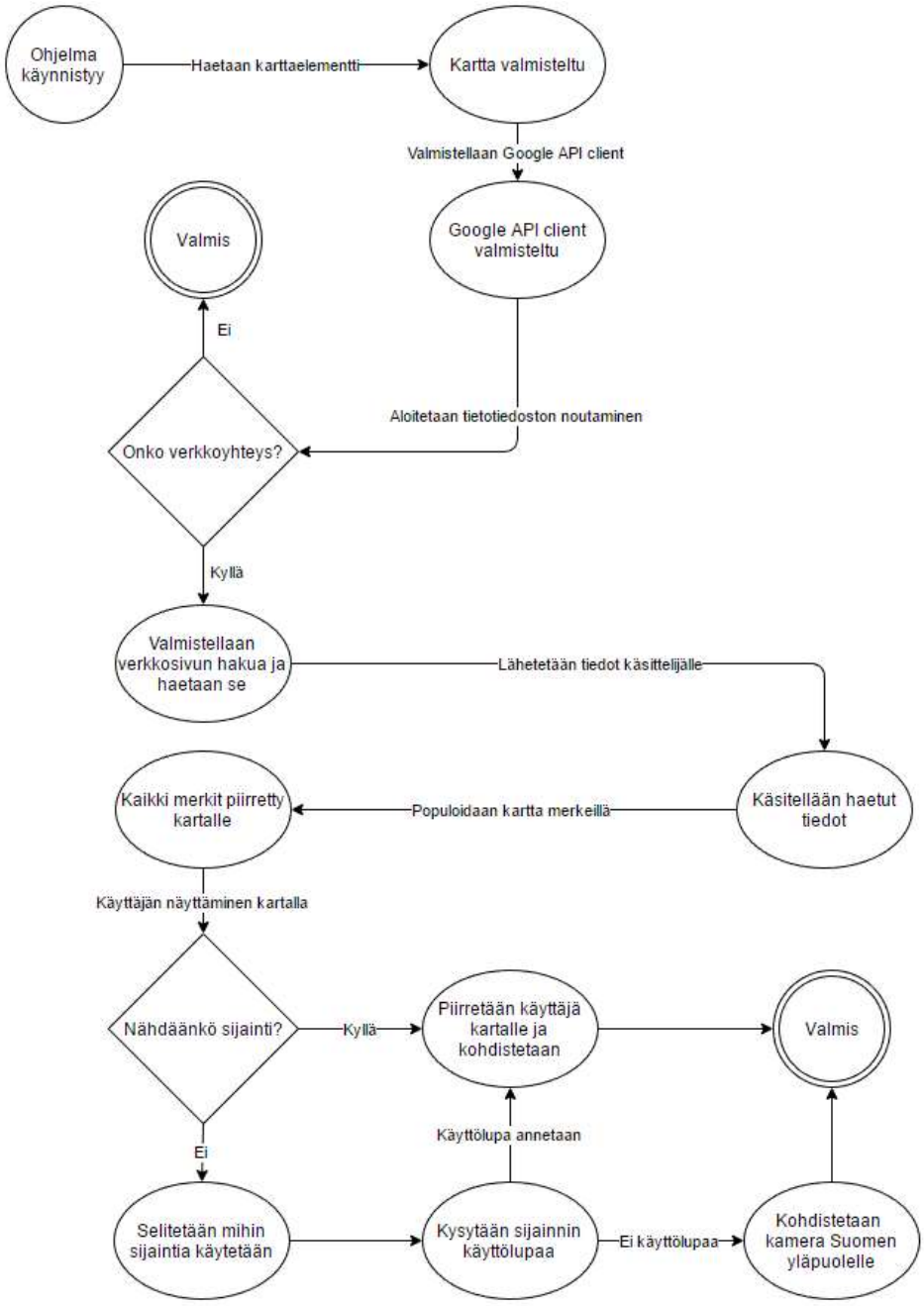
yhden korttelin, eli noin 100 metrin tarkkuudella. Koska ohjelmaa käytetään kaupunkien keskustoissa, ja on tärkeää nähdä käyttäjän sijainti yksittäisillä kaduilla, joudutaan käyttämään enemmän akkua ja prosessoria kuluttavaa FINE_LOCATION-tarkkuutta. Nyt käyttäjän sijainti nähdään ja sitä voidaan hyödyntää, jos käyttäjä on antanut tähän luvan, joten viimeiseksi kerrotaan vielä karttakomponentille, että sen pitää piirtää käyttäjän sijainti. Se onnistuu helposti karttaan rakennetulla lipulla, eli asetetaan kartan setMyLocationEnabled todeksi (true). (Google Developers, 2017-e.) Nyt kartalla näkyy automaattien sijaintien lisäksi myös käyttäjän sijainti, jos sijainnin lukeminen on vain sallittu. (Kuva 12.)



Kuva 12. Käyttäjä näkyy kartalla sinisenä pisteenä.

On kuitenkin mahdollista, että käyttäjä ei halua jakaa omaa sijaintiaan ohjelmalle, ja se voi johtua siitä, että käyttäjä ei tiedä mihin sijaintitietoja käy-

tetään. Siispä ohjelman käynnistyessä, kun tarkistetaan, saadaanko käyttäjän paikkatietoja nähdä, pitää luvan puuttuessa sitä pyytää. Androidin versiosta 6.0 eteenpäin lupien pyytäminen on onnistunut ohjelmaa ajettaessa. Tämän ansiosta voidaan siis pyytää lupaa nähdä käyttäjän sijainti, ja jos käyttäjä ei sitä halua jakaa, voidaan näyttää selitys sijaintitietojen käyttämiselle. Jos käyttäjä ei tämänkään jälkeen halua antaa ohjelman nähdä sijaintiaan, ei lupaa tarvitse kysyä enää uudestaan (Kuva 13). Käyttäjä voi tosin halutessaan antaa luvan myöhemmin ohjelman asetuksista.



Kuva 13. Ohjelman toimintajärjestys tässä vaiheessa.

3.6 Jo lisättyjen toimintojen kehitystä

Nyt kaikki ohjelman ydinkomponentit ovat valmiita, joten voidaan ryhtyä lisäämään ja parantamaan ohjelman muita toiminnallisuuksia. Edellä mainittiin automaattien tietolistan hakemisen ja käytön optimoinnista. Nyt ohjelma toimii siis niin, että valmistellaan karttakomponentti, tutkitaan, saadaanko nähdä käyttäjän sijaintia, ja sitten piirretään kartalle kaikki mitä pystytään, eli automaatit ja mahdollisesti käyttäjä.

Vaikka tietolista ei ole suuri, sen hakemiseen menee jonkin aikaa aina ohjelman käynnistyessä. Sen lisäksi, että ohjelman käynnistys viivästyy, se kuluttaa asiakkaan serverin resursseja. Siispä muutetaan listan käsittelyä niin, että se tallennetaan käyttäjän laitteelle. Kun ohjelma käynnistyy, se tarkistaa onko käyttäjän laitteessa jo tietolistaa tallennettuna, ja jos on, ei sitä tarvitse noutaa uudestaan, ellei se ole liian vanha. Jos listaa ei ole, tai se on liian vanha, se noudetaan verkosta uudestaan. Tällöin listaa ei tarvitse hakea joka käynnistyksen yhteydessä, mikä säästää sekä serverin resursseja, jolla tietolista on, että asiakkaan verkkoyhteyttä ja parantaa ohjelman käynnistymiseen kuluvaa aikaa.

Kun tietolista on haettu verkosta, se halutaan tallentaa puhelimen muistiin. Annetaan tietolistalle tallennettaessa nimi, ja kun ohjelma käynnistetään seuraavan kerran, voidaan tarkistaa tiedoston viimeisestä muokkauspäivämäärästä, onko se liian vanha käytettäväksi. Jos tiedoston on päivitetty tarpeeksi vähän aikaa sitten, käytetään sitä, ja näin säästetään datasiirrossa sekä käyttäjän että asiakkaan päissä. Jos tiedosto on liian vanha, haetaan uusi lista ja korvataan vanha tiedosto uudella. Vanhoja tietolistoja ei siis pidetä turhaan laitteen muistissa, vaan ne korvataan aina uusimmalla listalla, eikä täytetä laitteen muistia turhaan vanhoilla listoilla.

Sen lisäksi, että automaattien sijainnit piirtyvät kartalle, on tärkeää nähdä myös niiden tarkka katuosoite, aukioloajat, sekä automaatin tarjoamat palvelut (esim. onko rahan talletus mahdollista). Nämä tiedot on jo talletettu, mutta merkeissä oletuksena oleva infoikkuna ei tue monirivistä kuvauskenttää. Tämän vuoksi merkissä näkyy siis vain automaatin sijaintirakennus, sekä katuosoite, mutta muut tiedot leikkautuvat pois. Merkeille on siis luotava oma infoikkuna.

Ensiksi luodaan infoikkunan tyylitiedot sisältävä XML-tiedosto. Tähän tulee kaksi TextView-rakennetta, joista ensimmäinen sisältää otsikon, ja toinen kuvaustekstin. Näihin voidaan tehdä vapaasti omia muutoksia, kuten esimerkiksi lisätä kuvaukseen jokin kuva, tai muuttaa infoikkunan taustan väriä. Muutetaan otsikkotekstin fontti lihavoiduksi selkeyden parantamiseksi. Jotta uusi infoikkuna saadaan merkkien käytettäväksi, pitää luoda oma infoikkuna-adapteri. Se periytyy GoogleMap.InfoWindowAdapter-komponentista, ja toimiakseen tarvitseekin muutaman funktion. Näitä

ovat getInfoWindow, sekä getInfoContents. Kun infoikkuna halutaan näkyviin, kutsutaan näistä ensimmäistä, getInfoWindow. Se voidaan asettaa sisältämään kaiken infoikkunan tyylitiedon, jolloin voidaan kustoimoida esimerkiksi ikkunan kokoa tai taustaväriä. Tämä ei kuitenkaan ole tässä tapauksessa tarpeellista, joten asetetaan getInfoWindow palauttamaan null. Vain kun se palauttaa nullin, kutsutaan getInfoContents-funktiota (Kuva 14). Tässä tapauksessa ikkunan tyylitiedot haetaan normaalista infoikkunasta, mutta sen sisältämät tekstit voidaan kustomoida. Asetetaan otsikoksi ja kuvaustekstiksi klikatusta merkistä saatava otsikko ja kuvausteksti ja palautetaan ne. Tällöin ne näytetään merkin infoikkunassa. Lopuksi kerrotaan sekä kartalle, että merkkien yhdistelijälle, että halutaan käyttää uutta infoikkuna-adapteria. (Google Developers, 2016.)

```
private class MyCustomAdapterForItems implements GoogleMap.InfoWindowAdapter {
    private final View myContentsView;

    MyCustomAdapterForItems () {
        myContentsView = getLayoutInflater().inflate(R.layout.custom_infowindow, null);
    }

    @Override
    public View getInfoWindow (Marker marker) { return null; }

    public View getInfoContents (Marker marker) {
        TextView tvTitle = ((TextView) myContentsView.findViewById(R.id.txtTitle));
        TextView tvSnippet = ((TextView) myContentsView.findViewById(R.id.txtSnippet));

        tvTitle.setText(clickedClusterItem.getTitle());
        tvSnippet.setText(clickedClusterItem.getSnippet());

        return myContentsView;
    }
}
```

Kuva 14. Infoikkunassa getInfoWindow() palauttaa tyhjän (null), kutsutaan getInfoContents(), joka palauttaa vain infoikkunan tekstit, ilman tyylitietoja.

Ohjelmaa ensimmäistä kertaa käyttävä ei välttämättä ymmärrä heti, mitä kaikkea taustalla tapahtuu. Koska merkkien piirtämiseen menee hetki ohjelman käynnistymisen jälkeen, olisi hyvä kertoa käyttäjälle, että nyt pitää odottaa hetken aikaa, ettei käyttäjä suotta sulje ohjelmaa. Androidissa lyhyiden tekstien näyttämiseksi on luotu toast-toiminto. Toast on pieni tekstikupla, joka tulee ruudun alareunaan, ja siihen voidaan kirjoittaa mikä vain haluttu teksti, ja määritellä se näkyväksi halutuksi ajaksi. Kun tietolistaa haetaan ja jäsennellään, olisi hyvä näyttää käyttäjälle pieni teksti joka kertoo, että ohjelma ei ole jumissa, kuten esimerkiksi "odota hetki" tai "tietoja haetaan".

Toastin käyttö onnistuu seuraavasti: aluksi haetaan toastille ohjelman konteksti, sitten luodaan toast konteksti ja haluttu teksti yhdistämällä, ja lopuksi toast näytetään. Halutut tekstit säilötään eri muuttujiin, jolloin ne kaikki löytyvät samasta paikasta ja niitä voi tarvittaessa käyttää useammassa eri tilanteessa. Ohjelmaa testattaessa huomasin, että kun tietolista haetaan laitteen muistista, merkit piirtyvät kartalle hyvin pienellä viiveellä. Kun taas tietolista joudutaan noutamaan verkosta, on viive huomattavasti pidempi. Tämän vuoksi toast kannattaakin näyttää vain silloin, kun tietolista haetaan verkosta. Näihin voivat vaikuttaa käyttäjän internetyhteyden nopeus sekä laitteen suorituskyky. Ohjelman toimintakykyä muilla laitteilla testataankin myöhemmin, joten tämä asia merkitään muistiin myöhempää testausta varten. (Android Developers, 2017-i, 2017-j.)

3.7 **Valikkopalkki**

On huomioitava, että kirjoitushetkellä pankkiautomaatteja on Suomessa melkein 1400 kappaletta. Suurin osa näistä on normaaleja Otto-automaatteja, eli niissä onnistuu setelirahan nosto, sekä saldon ja tilitapahtumien tarkastelu. Joskus eteen voi tulla kuitenkin muita käyttövaatimuksia, kuten rahan talletus, tai liikuntarajoitteisuus. Tämän vuoksi normaalien Otto-automaattien lisäksi on olemassa esim. T-automaatteja (talletus), tai EP-automaatteja (esteetön, matala asiointikorkeus, sekä puhetuki, jolloin niitä voi käyttää myös esimerkiksi pyörätuolissa olevat henkilöt). Kartalla näkyy toistaiseksi kaikki automaatit kahdella eri merkillä: sininen merkki kertoo, että automaatilla onnistuu myös rahan talletus, kun taas oranssi merkki ilmaisee pelkän rahan nostamisen olevan mahdollista. Tämän vuoksi, jos käyttäjä haluaa tallettaa esimerkiksi kolikoita, pitää hänen klikkailla läpi kaikki lähialueen siniset merkit nähdäkseen, onko kyseessä O+-automaatti, vai T-automaatti, joista vain jälkimmäisessä onnistuu kolikoiden talletus. Paras ratkaisu ajan säästämiseksi ovat erilaiset filtterit.

Automaatit ovat jaettu tietolistassa viiteen eri kategoriaan: O, P, EP, T, ja O+. Käytännöllisintä onkin siis luoda viisi filtteriä, yksi jokaiselle automaatille, joista käyttäjä saa valita aktiiviseksi haluamansa. Aktiivinen filtteri tarkoittaa tässä tapauksessa sitä, että sen tyyppiset automaatit piirretään kartalle näkyviin. Oletuksena kaikki filtterit olisivat aktiivisia ja käyttäjä voisi epäaktivoida niistä haluamansa, esimerkiksi kolme ensimmäistä, jos haluaa nähdä vain ne automaatit, joissa setelirahan talletus on mahdollista.

Filtterien lisäksi olisi ohjelmassa hyvä olla joitakin käyttöohjeita vastaamaan mahdollisiin kysymyksiin ilman, että käyttäjän on otettava yhteyttä ohjelman tekijöihin. Tällaisia ohjeita ovat esimerkiksi sijaintitietojen käyttö, filtterien tarkoitus, eri automaattien tietojen selventäminen, sekä yhteydenottotiedot. Ohjelmalla on siis selkeä tarve valikolle, joka sisältäisi kaksi vaihtoehtoa: filtterit, ja lisätietoja/ohjeet.

Valikot, eli menut, ovat olennainen osa kaikenlaisten ohjelmien käyttöliittymiä. Siksi Android tarjoaakin tehokkaan kokonaisuuden erilaisten valikoiden luomiseen. Ohjelman oletusnäkymä on jaettu kahteen osaan: pieni yläpalkki, jossa on ohjelman nimi ja eri valikkonapit, sekä lopun ruudun täyttävä karttakomponentti (Kuva 4). Koska valikot voidaan jakaa kahteen osaan (filtterit ja lisätietoja), ei niitä kannata laittaa yhdessä yhden napin alle, vaan jättää ne erillisiksi napeiksi valikkopalkkiin. Näin säästetään käyttäjältä yksi tarpeeton klikkaus, ja tehdään käytöstä suoraviivaisempaa. Aloitetaan luomalla molemmat napit. Se onnistuu uudella XML-tiedostolla, joka sisältää kaksi kenttää. Kentät määritellään koko ajan näkyviksi, jolloin ne eivät yhdisty tarpeettomasti minkään toisen napin alle niin kauan, kuin ruudun leveys vain riittää. Napeille määritetään omat takaisinkutsufunktiot (callback-funktiot), joita kutsutaan nappia painettaessa. (Android Developers, n.d.-k.)

Nyt voidaan luoda napeille omat toiminnallisuudet ohjelmakoodiin. Filtterit ovat siis lista vaihtoehtoja, josta voidaan valita aktiiviseksi niin monta kuin halutaan, eli käytetään monivalintaruutuja (checkbox). Normaalisti valikot toimivat niin, että kun jotain valikon vaihtoehtoa painetaan, valikko menee pois. Monivalinnan tarpeen vuoksi tätä ei kuitenkaan voida käyttää filtterien näyttämiseksi, sillä aina kun käyttäjä valitsisi yhden filtterin, filterilista menisi pois näkyvistä, ja pitäisi avata uudelleen. (Android Developers, n.d.-k.) Siksi luodaankin filttereille valikkoikkunan sijaan oma dialogi-ikkuna, joka pysyy näkyvissä siihen asti, kunnes käyttäjä joko hyväksyy tekemänsä muutokset (onDialogPositiveClick), hylkää tekemänsä muutokset (onDialogNegativeClick), tai koskettaa jotain ruudun aluetta dialogi-ikkunan ulkopuolella, jolloin ikkuna menee pois ja tehdyt muutokset myös hylätään (Android Developers, n.d.-l).

Dialogi-ikkunaa käytettiin aikaisemmin myös näyttämään käyttäjälle teksti, jossa selitetään, mihin käyttäjän paikkatietoja ohjelmassa käytetään. Kun käyttäjä hyväksyy tekemänsä filtterimuutokset, tallennetaan uusi filterilista muistiin, poistetaan kaikki kartalla olevat merkit, ja piirretään ne uudelleen niin, että vain ne merkit jotka käyttäjä haluaa nähdä, piirretään. Kun filtteridialogi taas esitetään käyttäjälle, haetaan muistista käyttäjän aktivoimat filtterit, ja merkitään ne valintaruudut aktiivisiksi, jotta käyttäjän ei aina tarvitse valita kaikkia haluamiaan filttereitä uudestaan, ja muutosten tekeminen on helpompaa. Filtterit eivät kuitenkaan vielä tallennu niin, että samat olisivat käytössä, kun ohjelman sammutetaan kokonaan ja käynnistetään uudelleen. Nyt toinen valikon napeista toimii oikein, joten on aika siirtyä toiseen.

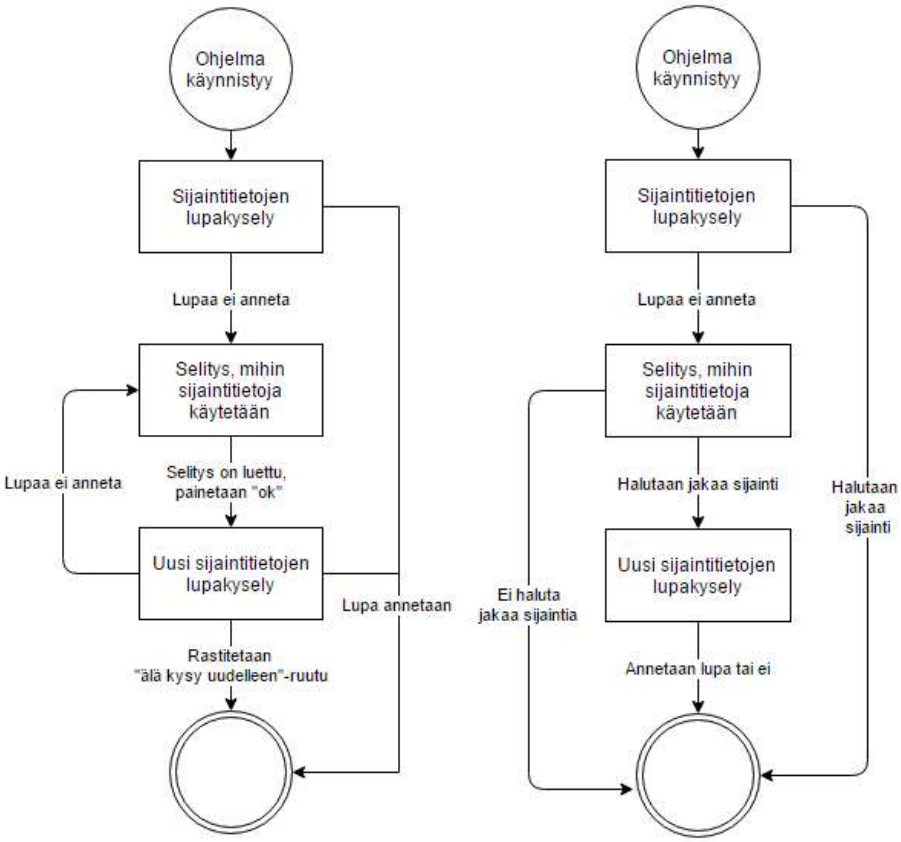
Valikon jälkimmäinen nappi on infonappi, eli sen takaa löytyy kaikenlaista lisätietoa ohjelmasta. Tässä listassa halutaan näyttää neljä eri nappia: lisätietoja ohjelmasta, lisätietoja filttereistä, lisätietoja sijainnin käytöstä, sekä yhteystiedot. Tämä tarkoittaa sitä, että lisätietoja-napin pitää synnyttää alleen pudotusvalikko, josta voidaan valita yksi neljästä edellä mainitusta vaihtoehdosta. Luodaan siis uusi XML-tiedosto joka sisältää tämä alivalikon

tiedot. Jokainen neljästä napista avaa oman dialoginsa, joissa on eri tekstit selittämään lisää ohjelmasta. Nämä dialogit ovat vain tekstiä, eivätkä vaikuta ohjelman toimintaan, joten dialogien päättämiseksi niihin lisätään vain yksi nappi, jossa voi lukea esimerkiksi "ok" tai "selvä", jotta käyttäjä saa dialogin helposti pois ruudulta sen luettuaan. (Android Developers, n.d.-k.)

3.8 **Käyttäjän tekemien muutosten tallentaminen**

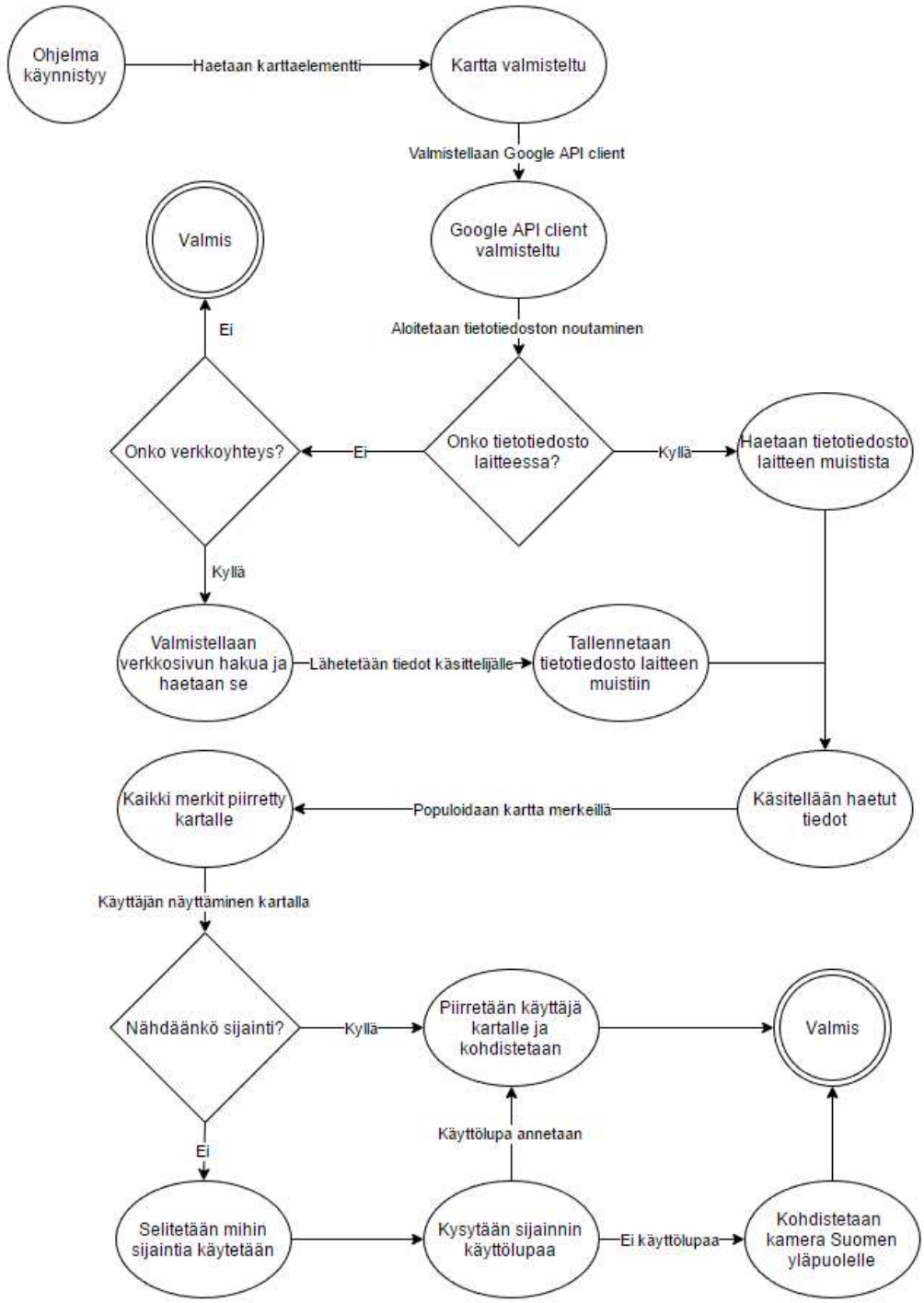
Android-sovelluksissa istuntokohtaiset muutokset nollautuvat usein, esimerkiksi näytön sammuessa, ohjelman joutuessa taustalle, näytön orientaation muuttuessa, ja tietenkin ohjelmaa sammutettaessa. Siksi onkin tärkeää tallentaa käyttäjän tekemät muutokset pysyvämpään muotoon, ja tähän tarjotaankin Preference- ja SharedPreferences-rajapintoja. SharedPreferences tallettaa ohjelmakohtaiseen tiedostoon listan avain-arvo-pareja, joita voidaan tallentaa käyttäjän tietämättä, tai käyttäjän tekemien päätösten seurauksena. Ohjelman käynnistyessä voidaan tiedosto noutaa laitteen muistista, ja lukea sieltä halutut arvot. Tällä hetkellä ainoa käyttäjältä kysyttävä asia on paikkatietojen käyttölupa, jota kysytään ohjelmaa ensimmäisen kerran käynnistettäessä. Jos käyttäjä ei anna lupaa, näytetään selitys mihin paikkatietoja käytetään. Tämän jälkeen käyttäjältä kysytään lupaa uudestaan, jolloin käyttäjän pitää rastittaa ruutu "älä kysy uudelleen", jos hän ei halua nähdä kysymystä enää toiste. Prosessi on liian pitkä ja sekava käyttäjille, jotka eivät halua jakaa sijaintiaan. Tämän vuoksi muutetaan sitä niin, että käyttäjä voi jo selityksen luettuaan päättää, haluaako antaa sijaintitietojen käyttöluvan. Tällöin vältytään yhdeltä turhalta lisäaskeleelta, ja ohjelman aloitus on vähemmän häiritsevä.

Ohjelmaa aloitettaessa noudetaan SharedPreferences-rajapinnan avulla avain-arvo-parit. Jos tiedostoa ei ole vielä olemassa, sitä ensimmäistä kertaa haettaessa se luodaan. Lisätään tiedostoon yksi avain, jonka arvo tallennetaan sen perusteella, mitä käyttäjä vastaa alussa näytettävään lupakyselyyn. Kun lupakyselyyn on vastattu, tallennetaan vastaus tiedostoon ja edetään normaalisti, joko ilman käyttäjän sijaintitietoja tai niiden kanssa. Nyt lupakyselyitä ei suoriteta aina ohjelman käynnistyessä, jos käyttäjä ei ole antanut lupaa käyttää sijaintitietoja. (Kuvat 15, 16 ja 17.)



Kuva 15. Vanha lupakysely vasemmalla. Huomioi mahdollinen uudelleenkyselyn aiheuttama loppumaton kehä, jos käyttäjä ei rastita ”älä kysy uudelleen”-ruutua.

Kuva 16. Uusi lupakysely oikealla. Uudessa metodissa ei ole mahdollista loppumatonta kyselyä, ja se on helpommin ymmärrettävissä päivitettyjen tekstien ansiosta.



Kuva 17. Ohjelman julkaisuversion toimintajärjestys.

3.9 **Mainosbanneri**

Jäljellä on enää viimeinen asiakkaan toivoma ominaisuus, eli pieni mainosbanneri ohjelmaan. Mainosten lisääminen Android-sovelluksiin on tehty erittäin helpoksi Googlen Firebasen avulla. Asiakkaalla on käyttäjätunnukset Firebase-järjestelmään, johon hän on lisännyt projektin ja saanut pro-

jektikohtaisen avaimen. Firebase tarjoaa kattavat ohjeet mainosten lisäämiseen omilla sivuillaan, joten niitä seuraten työn saa suoritettua pikaisesti. Projektiin tuodaan ensimmäiseksi Mobile Ads SDK -paketti, ja projekti synkronoidaan. Sitten ohjelmakoodiin määritellään ohjelman käynnistykseen initialisointikäsky, jonka toinen argumentti on projektikohtainen Firebase-avain. Testausvaiheessa käytetään Firebasen tarjoamaa testausavainta, jota käytettäessä oikeiden mainosten sijaan palvelu tarjoaa vain testimainoksia, jotka ovat oikeiden mainosten kokoisia. Näiden avulla voidaan rakentaa ohjelman ulkoasua ja testailla mainosten toimivuutta, kuitenkaan oikeita mainoksia käyttäen, sillä Google on ehdottomasti kieltänyt kehittäjiä avaamasta omassa sovelluksessaan näkyviä mainoksia, ja sen tekeminen saattaa johtaa tunnusten jäädytykseen. Siksi turvallisinta onkin käyttää tarjottua testiavainta, jos sattuu vahingossa avaamaan mainoksen ohjelmassaan. (Firebase, 2017-a.; Firebase, 2017-b.)

Kun ohjelma on valmisteltu tukemaan mainoksia, pitää valita minkälaisia mainoksia haluaa näyttää, ja mihin ne ohjelman käyttöliittymässä haluaa sijoitta. Käyttöliittymä koostuu valikkopalkista ja koko loppuruudun peittävästä kartasta. Siistein ja vähiten tiellä olevin paikka mainoksille onkin siis kartan alapuolella. Koko ruudun levyinen, mutta korkeudeltaan matala bannerimainos on tähän sopiva valinta. Eri mainosten layout-elementit ovat tarjolla Firebasen ohjeissa. Haetaan bannerimainoksen elementti ja lisätään se ohjelman layout-tiedostoon karttaelementin alapuolelle. Säädetään karttaelementin korkeutta vielä niin, että mainos ei peitä karttaelementin alaosaan ilmestyviä, lisätoiminnallisuuksia tarjoavia nappeja. Lopuksi lisätään ohjelmakoodiin mainosten initialisoinnin jälkeen kutsupyyntö mainokselle, ja yhdistetään se juuri lisättyyn elementtiin. Näin ohjelman käynnistyessä luodaan samalla yhteys mainosjärjestelmään verkon yli, ja sieltä tarjotaan mainos ohjelmalle, joka sijoitetaan mainoselementtiin. Koska asiakas haluaa tarjota ohjelmasta ilmaisen ja maksullisen version, voidaan tässä kappaleessa tehdyt lisäykset jättää tekemättä maksulliseen versioon. (Google Developers, 2017-f.)

4 YHTEENVETO JA JATKOKEHITYS

Ohjelman tekoon ryhdyttäessä olivat päätavoitteet ja -työkalut selkeät. Suurimmat ongelmat syntyivät Java-ohjelmointikielen vieraudesta ja layout-rakenteiden tekemisestä XML-merkintäkielellä. Eri käytettyjen ominaisuuksien dokumentaatio oli erittäin kattavaa, kiitos Googlen ahkeruuden, ja moniin ongelmiin niiden käytön suhteen löytyikin ratkaisu foorumeilta ja dokumentaatiosta, joten kompromisseja ominaisuuksien tai niiden implementoinnin suhteen ei tarvinnut tehdä. Ohjelma saatiin valmiiksi aikataulussa, ja asiakas oli tyytyväinen, sillä kaikki tavoitteet saavutettiin ja kehityksen aikana syntyi jopa muutama lisäidea käyttömukavuuden paran-

tamiseksi. Käyttötestaus oli ainoa asia, joka jouduttiin karsimaan aikataulusta, mutta pyrin kehittämään käyttöliittymän parhaani mukaan uusia käyttäjiä ajatellen.

Ohjelmasta on nyt versio 1.0 valmiina ja kuten aina ohjelmistokehityksessä, jatkokehitykselle on tilaa. Android 7.0 julkaistiin ohjelman kehityksen aikana, joten sen mukanaan tuomat lisäominaisuudet ja rajoitteet olisi hyvä käydä läpi versioon 1.1. Käyttötestausta olisi myös hyvä sisällyttää seuraavaan vaiheeseen, jotta mahdolliset puutteet ja epäselvyydet saataisiin myös parannettua versioon 1.1. Yksi asia, joka rajattiin heti alussa pois, oli myös tablettikokoisen näytön käyttöliittymät, joten käyttöliittymän ulkoasu tableteilla kannattaisi myös testata ennen seuraavan version julkaisua. Versioon 1.2 ehdottaisin virheenkeston kehitystä, sillä ohjelma vaatii uudelleenkäynnistyksen esimerkiksi silloin, kun se käynnistetään ilman internetyhteyttä, ja yhteys palautuu ohjelman jo käynnistyttyä. Yritin kehityksessä huomioida mahdollisia nopeusongelmia, ja tässä auttoi ainakin omassa laitteessani oleva hidas verkkoyhteys. Pystyin tämän avulla toteamaan, että tietolistan nouto ei ole liian hidasta. Siitä voisi saada nopeamman esimerkiksi jakamalla automaatit asiakkaan serverillä erilaisiin ryhmiin, esimerkiksi maakuntien, vanhojen läänien, tai vain harkiten rajattujen alueiden mukaan, jolloin esimerkiksi Helsingissä olevat käyttäjät voisivat määrittää haluavansa ladata vain Helsingin alueen automaatit. Suomessa verkkoyhteyksien nopeus mobiililaitteilla on kuitenkin huonoimmassakin tapauksessa tarpeeksi hyvä, ja tietotiedoston koko on niin pieni, että tällaista automaattien jakoa ei tarvinnut tehdä ensimmäiseen versioon. Seuraaviin versioihin voisi myös lisätä kameran sijainnin talletuksen, jolloin kamera siirtyisi siihen, mihin käyttäjä on sen viimeksi jättänyt, vaikka ohjelma sulkeutuisi välissä, ja karttaan voisi samalla lisätä napin, josta kamera keskittyisi taas käyttäjän sijaintiin. Ohjelma on nyt kuitenkin hyvässä pisteessä; se toimii nopeasti ja varmasti, se on kevyt käyttää ja helppo ymmärtää, se kestää muutoksia, ja kiitos Androidin yhteensopivuuden, sitä voi käyttää valtaosassa Suomessa olevia Android-puhelimia.

Työ oli mukava tehdä, ja se oli opettavainen kokemus. Olen suorittanut koulussa yhden kurssin mobiiliohjelmointia, joten onneksi ei tarvinnut aloittaa projektia aivan tyhjältä pöydältä. Projektista opin paljon Android-sovellusten kehittämisen lisäksi myös Java-kielestä, sekä tietenkin olio-ohjelmoinnista. Suosittelisinkin siis kaikkia Android-sovellusten kehittämisestä kiinnostuneita lähteä yrittämään. Android Studio helpottaa työskentelyä suuresti, ja hyvä dokumentaatio sekä massiivinen käyttäjä- ja kehittäjäkunta takaavat vastausten löytymisen kysymykseen kuin kysymykseen.

LÄHTEET

Android Developers, n.d.-a, *Add C and C++ Code to Your Project*. Noudettu 04.02.2017 osoitteesta https://developer.android.com/studio/projects/add-native-code.html.

Android Developers, n.d.-b, *Requesting Permissions*. Noudettu 04.02.2017 osoitteesta https://developer.android.com/guide/topics/permissions/requesting.html.

Android Developers, n.d.-c, *ConnectivityManager*. Noudettu 04.02.2017 osoitteesta https://developer.android.com/reference/android/net/ConnectivityManager.html.

Android Developers, n.d.-d, *AsyncTask*. Noudettu 05.02.2017 osoitteesta https://developer.android.com/reference/android/os/AsyncTask.html.

Android Developers, n.d.-e, *HttpUrlConnection*. Noudettu 05.02.2017 osoitteesta https://developer.android.com/reference/java/net/HttpURLConnection.html.

Android Developers, n.d.-f, *Getting the Last Known Location*. Noudettu 17.02.2017 osoitteesta https://developer.android.com/training/location/retrieve-current.html.

Android Developers, n.d.-g, *Markers*. Noudettu 04.03.2017 osoitteesta https://developers.google.com/maps/documentation/android-api/marker.

Android Developers, n.d.-h, Getting the Last Known Location. Noudettu 25.03.2017 osoitteesta https://developer.android.com/training/location/retrieve-current.html#last-known

Android Developers, n.d.-i, *Toasts*. Noudettu 06.04.2017 osoitteesta https://developer.android.com/guide/topics/ui/notifiers/toasts.html

Android Developers, n.d.-j, *Toast*. Noudettu 06.04.2017 osoitteesta https://developer.android.com/reference/android/widget/Toast.html

Android Developers, n.d.-k, *Menus*. Noudettu 07.04.2017 osoitteesta https://developer.android.com/guide/topics/ui/menus.html

Android Developers, n.d.-l, *Dialogs*. Noudettu 08.04.2017 osoitteesta https://developer.android.com/guide/topics/ui/dialogs.html

Automatia Pankkiautomaatit Oy, 2017, *Otto – Ottopisteiden kotisivu*. Noudettu 02.02.2017 osoitteesta https://otto.fi/.

Firebase, 17.05.2017-a, *Add Firebase to Your Android Project*. Noudettu 20.05.2017 osoitteesta https://firebase.google.com/docs/android/setup

Firebase, 17.05.2017-b, Get Started in Android Studio. Noudettu 20.05.2017 osoitteesta https://firebase.google.com/docs/admob/android/quick-start

Google, 2017-a, *Android Platform/API-version Distribution*. Noudettu 02.02.2017 ohjelmasta Android Studio.

Google Developers, 21.03.2017-a, *Selectively compiling APIs into your executable*. Noudettu 24.04.2017 osoitteesta https://developers.google.com/android/guides/setup.

Google Developers, 21.03.2017-b, *Release Notes February 2017 – v. 10.2*. Noudettu 24.04.2017 osoitteesta https://developers.google.com/android/guides/releases#february_2017_-_v102.

Google Developers, 14.03.2017-c, *Google Maps Android Marker Clustering Utility*. Noudettu 25.04.2017 osoitteesta https://developers.google.com/maps/documentation/android-api/utility/marker-clustering.

Google Developers, 21.03.2017-d, *Accessing Google APIs*. Noudettu 25.04.2017 osoitteesta https://developers.google.com/android/guides/api-client

Google Developers, 14.03.2017-e, Location Data. Noudettu 21.03.2017 osoitteesta https://developers.google.com/maps/documentation/android-api/location

Google Developers, 16.07.2017-f, Banner Ads. Noudettu 20.05.2017 osoitteesta https://developers.google.com/admob/android/banner

Google Developers, 09.11.2016, GoogleMap.InfoWindowAdapter. Noudettu 27.03.2017 osoitteesta https://developers.google.com/android/reference/com/google/android/gms/maps/GoogleMap.InfoWindowAdapter

Wikipedia, 22.04.2017-a, *Google Play Services*. Noudettu 24.04.2017 osoitteesta https://en.wikipedia.org/wiki/Google_Play_Services.

Wikipedia, 07.06.2017-b, JSON. Noudettu 09.06.2017 osoitteesta https://en.wikipedia.org/wiki/JSON.

Xamarin, 2017, *Share Code Everywhere*. Noudettu 23.04.2017 osoitteesta https://www.xamarin.com/platform/.